Druga część opisu mikrokomputera APPLE poświęcona została oprogramowaniu. Tym razem jednak rzecz nie w opisie fantastycznych gier komputerowych czy programów graficznych — pozwalających architektowi zaprojektować dom od dachu po fundamenty z uwzględnieniem kształtu krzeseł w jadalni. Aby realizacja wspomnianych programów była możliwa w mikrokomputerach, stosowany jest program zarządzający o nazwie MONITOR (lub SYSTEM OPERACYJNY, gdy obsługuje również współpracę z dyskami). W jego skład wchodzi szereg procedur, od stopnia wyrafinowania których zależy prostota (a często i możliwości) programów użytkowych.

Procedury realizujące zbliżone funkcje można obecnie spotkać w oprogramowaniu zasadniczym większości „szanujących się" mikrokomputerów. Tym, którzy kompletują program MONITOR dla komputera własnej konstrukcji, radzimy skorzystać z dobrze sprawdzonego wzorca.

apple

O jabłku opowieść (2)

Oprogramowanie

Opisując mikrokomputer APPLE II warto wspomnieć o dwóch wersjach oprogramowania podstawowego mikroprocesora 6502. Wersją pierwszą, nieco starszą jest SMR (System Monitor ROM). W wersji tej dostępny jest język INTEGERBASIC (interpreter języka BASIC z ograniczeniem do liczb całkowitych) oraz MINI-ASSEMBLER 6502. Drugą wersją jest system AMR (Autostart Monitor ROM). Obok znacznie bogatszego oprogramowania podstawowego, językiem wysokiego poziomu jest tu APPLESOFT.

A oto podstawowa struktura oprogramowania umieszczonego w pamięci ROM dla AMR. Zarządcą i zarazem niewolnikiem systemu jest program SYSTEM MONITOR. Kontroluje on realizację wszystkich programów, tłumaczy niektóre na język wewnętrzny mikroprocesora 6502, ale też wszystkie one przekazują mu do wykonania różne polecenia.

SYSTEM MONITOR AMR zawiera między innymi mini DISASSEMBLER (mini — ponieważ nie potrafi przetwarzać adresów symbolicznych). Mini DISASSEMBLER uwzględnia 56 typów rozkazów i 13 trybów adresowania. Przekazywana informacja składa się z trzech części: adresu, kodu (w systemie szesnastkowym) oraz rozkazu. SYSTEM MONITOR składa się z szeregu procedur. Oto najważniejsze z nich:

Podprogram COUT (ang. Character OUTput) — wyprowadza znaki na urządzenia zewnętrzne w czterech postaciach: normal, inverse (negatyw), flash (migająca), esc-function (znak edytorski). Znaki te kodowane są w akumulatorze. Podprogram ten uwzględnia jedynie duże litery, ma pełną kontrolę nad kursorem, jego aktualnym położeniem na ekranie, organizuje tzw. okna, czyli liczbę kolumn i wierszy. Maksymalnie dopuszcza 40 kolumn w 24 wierszach. Program ten uruchamia w określonych warunkach głośnik, który generuje sygnał akustyczny o częstotliwości 1 KHz co 1/10 sekundy.

Podprogram COUT1 — wyprowadza znaki z akumulatora na ekran monitora. Może być on wywoływany przez COUT.

Podprogram SETINV (ang. SET INVerse) — uruchamiany bezpośrednio przez COUT. Przekształca obraz ekranu na jego negatyw.

Podprogram SETNORM (ang. SET NORMal) — powoduje powrót do pozytywu.

Podprogram CROUT (ang. Carriage Return OUTput) — współpracuje z urządzeniami peryferyjnymi, przesyłając do nich informację o funkcji RETURN kończącej dany rozkaz, czy linię.

Podprogram CROUT1 — współpracuje z podprogramem COUT dokonując odpowiednich korekt i wymazując niepotrzebne informacje z ekranu i pamięci.

Podprogram PRBYTE (ang. PRint BYTE) — kontroluje urządzenia wyjściowe; przesyła do nich zawartość akumulatora.

Podprogram BELL — generuje sygnał akustyczny poprzez urządzenie wejściowe.

Brak wyobraźni

Znajomy młody informatyk, do niedawna z irytacją reagujący na przejawy zniechęcenia wśród kolegów i demencji przełożonych, oświadczył nagle stanowczo, że cała praca w tej informatyce jest bezsensowna. Że być informatykiem — głupio; i że od dzisiaj liczą się dla niego tylko pieniądze. Pisać do INFORMATYKI nie będzie, dopóki gdzie indziej zapłacą więcej. Na argument, że mikroinformatyka zmienia sytuację, że nie całkiem zależna jest od resortowych „informatycznych" bankructw, odpowiadał z przekonaniem, że „mikroinformatyka pójdzie tą samą drogą". To jest — donikąd, niszczona skutecznie przez różne „całościowe" wizje i wytyczne.

Dzisiaj jest to już dość częsta postawa, wyrażająca z a s a d ę, będącą symptomem informatycznej aury. Postawa, która znajduje sobie wiele przekonywających uzasadnień, potrafiąca znaleźć poparcie w wielu liczbach, statystykach, prognozach. I choć może jest to na swój sposób postawa rozumna, to przecież jednocześnie — nie do przyjęcia.

Rezygnacja z aktywności może mieć dziś miejsce (i ma!) w każdej niemal sferze życia społecznego. Niszczy lub degeneruje społeczne funkcje, do jakich — w następstwie świadomego wyboru czy nie — jesteśmy powołani. Poddanie się apatii powoduje, że ta nasza nowa rola (przeczekiwacza) obezwładnia nas samych, wpycha w ślepy zaułek życia rodzinnego, towarzyskiego... Rodzi cynizm i zgorknienie.

Konsekwencja własnych poczynań w życiu publicznym wymaga dzisiaj sporego hartu, samodyscypliny, chroniącej przed mirażem tanich zasług i jałowych pochwał. I zapewne musi to być bardziej „na przekór" niż „w zgodzie" z otoczeniem. Ale tak czy inaczej ważniejszy jest wybór „za czymś" niż „przeciw czemuś". Owo „za" rozstrzyga o sensie własnych poczynań.

To nie agitacja. Chcę tylko zwrócić uwagę na możliwość daną przez — jeszcze horrendalnie drogą, ale już dostępną — mikroinformatykę. Że ze świadomością celu (także społecznego), a zyskawszy dogodne narzędzie, można — drążąc powoli — ruszać bryłę z posad...

Trzeba hartu, pewności swego. Trzeba umiejętności. I konsekwencji. Dopiero pogodzenie się ze słabością oznacza porażkę.

Ów młody informatyk znalazł się tam (także w sensie duchowym), gdzie dają więcej pieniędzy, gdzie może wyślą za granicę, dadzą awans. Jeśli nawet kosztem zawodowej i społecznej degradacji, to przecież — korzyści większe.

Dobrze przynajmniej, że nie udaje satysfakcji.

ZBIGNIEW GLUZA

Podprogram BELL1 — generuje sygnał akustyczny 1 kHz z głośnika komputera co 1/10 sekundy.

Podprogram RDKEY (ang. ReaD a KEY) — reaguje na wprowadzanie pojedynczych znaków z klawiatury; uwidocznia mały migający kwadracik, określający aktualną pozycję kursora kontrolowanego przez podprogram COUT. Podczas gdy RDKEY czeka na naciśnięcie klawisza, dodaje co 1/10 sekundy jedynkę do określonej pary komórek pamięci. Gdy kod znaku zostanie wprowadzony, wartości z obu komórek są dodawane tworząc liczbę z zakresu od 1 do 65 535. Ponieważ wynik jest zdarzeniem losowym, właściwość tę wykorzystuje się do tworzenia **generatora liczb losowych.**

Podprogram RDCHAR (ang. ReaD CHARacter) — współpracuje z RDKEY kontrolując jego działanie.

Podprogram KEYIN — odczytuje znaki z klawiatury. Po naciśnięciu klawisza, kursor na moment znika, a kod znaku zostaje przesłany do akumulatora.

Podprogram GETLIN (ang. GET a LINe) — pełni podobne funkcje do podprogramu RDKEY. W przeciwieństwie do niego, reaguje jednak nie na kod pojedynczego znaku, lecz na instrukcję RETURN. Specjalnym kodem informuje, w jakim języku aktualnie pracuje komputer. GETLIN współpracuje z podprogramami RDKEY oraz COUT i COUT1.

RDKEY przesyła kody z klawiatury do bufora w pamięci, skąd następnie zostaje on przepisany na ekran monitora. Wprowadzenie RETURN powoduje, że RDKEY wywołuje program GETLIN, a ten — ponownie RDKEY itd. Całością operacji steruje program COUT, odpowiednio sterujący kursorem i wyświetlaniem wprowadzonych znaków. GETLIN dopuszcza długość linii do 255 znaków. W przypadku przekroczenia tej granicy GETLIN kasuje całą linię, uruchamiając podprogram BELL1, generujący dźwięk o wysokości 1 kHz, po czym wraca do programu pierwotnego.

Podprogram WAIT — przerywa realizację programu na czas zależny od zawartości akumulatora. Zatrzymuje on wykonanie programu od 13 μs do 1/6 s.

Kolejne istotne podprogramy, charakterystyczne jednak dla specyficznej grafiki APPLE'a to: **SETCOL, NEXTCOL, PLOT, HLINE, VLINE, SCRN-SUBROUTINES.**

Organizacja SYSTEMU MONITOR AMR umożliwia znaczną jego rozbudowę przez dołączenie dodatkowych pamięci PROM i EPROM. Można zatem tworzyć własne podprogramy tego systemu.

Oprogramowanie dla APPLE'A zazwyczaj dostarczane jest na dyskietkach (5 1/4″ lub znacznie rzadziej 8″). Organizacja dyskietek 5 1/4″ polega na podziale dysku na 35 ścieżek (ang. tracks). Trzy z nich (najbadziej zewnętrzne) zajmuje na ogół sytem operacyjny DOS, który steruje współpracą z napędami dyskietek. Ścieżka środkowa rezerwowana jest na spis plików dostępnych na dysku. Dla użytkownika pozostaje zatem trzydzieści jeden wolnych ścieżek.

Każda ścieżka składa się z 16 bądź 13 sektorów, a jeden sektor umożliwia operowanie blokami o pojemności 256 bajtów. Użytkownik ma więc do dyspozycji 126 976 bajtów na każdej dyskietce. Bardzo pożyteczną procedurą (będącą częścią systemu operacyjnego) jest podprogram **RWTS** (ang. Read or Write a Track and Sector). Umożliwia on dostęp do dowolnego sektora dowolnej ścieżki.

```
5  REM  SITO ERATOSTENESA
6  REM  LICZBY PIERWSZE
10 HOME :INPUT "ILOSC LICZB PIERWSZYCH N=";N
14 PRINT CHR$ (12)
15 PRINT :GET A$:IF A$= "3"THEN PR# 3
20 DIM X(N):DIM Y(N)
30 X(0)= 3:Y(0)= 3:K= 1:V= 0
33 PRINT :PRINT :PRINT "2";" ";"3";
40 FOR I= 1TO N
44 : X(I)= X(I- 1)+ 2
50 : FOR L= 0TO K
55 : : UNLESS (X(I)/ Y(L)= INT (X(I)/ Y(L)))
58 : : : IF (Y(L)> SQR (X(I)))
60 : : : : V= V+ 1:L= V:K= V+ 1
66 : : : : Y(L)= X(I)
70 : : : : PRINT " ";Y(L);
77 : : : : GOTO 99
80 : : : :FIN
88 : : :NEXT L
90 : :FIN
99 :NEXT I
```

Na wydruku zaprezentowano próbkę programowania w języku APLUS. Program realizuje algorytm wyszukujący liczby pierwsze, znany pod nazwą „Sito Eratostenesa". Zamieszczone obok liczby otrzymano właśnie w wyniku realizacji programu. Strukturalne możliwości programowe w języku APLUS pozwalają na wyjątkowo przejrzyste przełożenie algorytmu na „język komputerowy". Nie znane są nam jednak implementacje tego języka dla komputerów innych niż produkowane przez firmę APPLE. Dlatego też nie podzielamy zgłoszonej nam przez autora tekstu opinii o potrzebie intensywnej popularyzacji tej właśnie odmiany języka BASIC. Prosimy jednak Czytelników o ewentualną korektę naszych przekonań. (Red.)

Na potrzeby APPLE'a stworzono wiele makroasemblerów, dostarczanych jako oprogramowanie dodatkowe na dyskach elastycznych. **MERLIN ASSEMBLER.** Makroasembler ten, napisany specjalnie na potrzeby APPLE'a, daje możliwość rozbudowy systemu redagowania tekstów, może współpracować z płytkami CARD-80 i pamięci 16 KB.

MICROSOFT'S ASSEMBLY LANGUAGE DEVELOPMENT SYSTEM. System ten składa się z trzech programów ASSEMBLER:
— ASSEMBLER 8080
— ASSEMBLER Z80
— 6502-kod.
Programy te można używać niezależnie bądź razem.

ALD SYSTEM II. Makroasembler dla APPLE'a napisany przez Paula Lutusa.

EDITOR/ASSEMBLER, Asembler ten ma bogate możliwości redagowania plików.

LISA ASSEMBLER, także autorstwa Paula Lutusa. Asembler ten umożliwia wykorzystanie całej gamy możliwości graficznych APPLE'a przez dołączanie procedur bibliotecznych. Stanowił on punkt wyjściowy dla propozycji graficznych w mikrokomputerze LISA.

```
]RUN
ILOSC LICZB PIERWSZYCH N=1000

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 7
9 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157 163
167 173 179 181 191 193 197 199 211 223 227 229 233 239 241 251
257 263 269 271 277 281 283 293 307 311 313 317 331 337 347 349
353 359 367 373 379 383 389 397 401 409 419 421 431 433 439 443
449 457 461 463 467 479 487 491 499 503 509 521 523 541 547 557
563 569 571 577 587 593 599 601 607 613 617 619 631 641 643 647
653 659 661 673 677 683 691 701 709 719 727 733 739 743 751 757
761 769 773 787 797 809 811 821 823 827 829 839 853 857 859 863
877 881 883 887 907 911 919 929 937 941 947 953 967 971 977 983
991 997 1009 1013 1019 1021 1031 1033 1039 1049 1051 1061 1063 1069
1087 1091 1093 1097 1103 1109 1117 1123 1129 1151 1153 1163 1171
1181 1187 1193 1201 1213
```

Najistotniejsze i największe z programów użytkowych, to cztery „mega-programy":

WORDSTAR to program do redagowania tekstów. W połączeniu z płytką CARD-80 daje możliwość formatowania tekstu bezpośrednio na ekranie (ang. ON-SCREEN-FORMATING). Program ten ma bogate możliwości redakcyjne, a nawet jest w stanie kontrolować swoje działanie, np. sprawdza, czy podział sylabiczny jest sensowny, podaje też użytkownikowi konkretne propozycje. Wymaga zastosowania systemu CP/M.

VISICALC pracuje w tabeli 63 kolumn oraz 254 wierszy. W każdym z 63 × 254 pól może być zakodowana albo litera, albo wartość arytmetyczna. Poszczególne pola mogą być ze sobą powiązane przez matematyczne formuły. Jeżeli w tabeli zmienia się dana wartość, wszystkie pozostałe zależne od niej wartości również odpowiednio się zmieniają. Jest to punkt wyjścia do konstruowania skomplikowanych modeli fizycznych (np. teoria pola), bez potrzeby układania skomplikowanych programów, czy też znakomite narzędzie dla ekonomisty. Program wymaga stosowania systemu operacyjnego CP/M.

MULTIPLAN ma wszystkie zalety programu VISICALC oraz dodatkowo możliwość konstruowania pól chronionych a także sortowania w zależności od posiadanej cechy. Program ten wymaga stosowania systemu operacyjnego CP/M.

PIE WRITER to program do redagowania tekstów w 80 kolumnach. Formatowanie tekstu możliwe jest w wariantach pisma blokowego oraz ozdobnego.

PIOTR TYMOCHOWICZ

Większość debiutujących użytkowników mikrokomputerów staje wobec problemu zdobycia oprogramowania. Jedną z dróg może być przenoszenie programów z dużych komputerów. Wprawdzie programy te (często napisane w FORTRANIE) nie działają zbyt efektywnie na komputerach wykorzystujących mikroprocesory 8-bitowe, lecz w warunkach domowych — mogą „liczyć się" choćby całą noc. Podejście takie jest na pewno bardziej rozsądne niż pisanie samemu wszystkiego od początku. Autorzy poniższego tekstu zamiast żmudnego przepisywania wydruków zrealizowali — kosztem pracy koncepcyjnej — rozwiązanie na miarę krajowych możliwości. W rezultacie przenieśli kilkadziesiąt procedur numerycznych z minikomputera PDP-11/45 na mikrokomputer TRS 80.

Warto jeszcze wspomnieć, że w ten sposób można przenosić programy wynikowe generowane przez kompilatory skrośne dużych maszyn cyfrowych.

Przenoszenie informacji z komputerów dużych na osobiste

Problem komunikacji komputera osobistego z dużą maszyną cyfrową rozwiązywany jest na ogół poprzez użycie modemu dołączonego do zwykłego aparatu telefonicznego. Tyle że w naszych warunkach jest to rozwiązanie nierealne i trzeba, niestety szukać innych dróg. Problem polega na tym, że duże komputery używają do przenoszenia informacji taśm magnetycznych lub co najwyżej dysków elastycznych 8". Nośniki te nie stanowią standardowego wyposażenia (ze względu na cenę) komputera domowego. Na szczęście wiele minikomputerów wyposażonych jest w pamięci kasetowe PK-1, które zapisują informacje na zwykłych kasetach magnetofonowych. Poniżej omówimy sposób odczytu informacji z takiej kasety.

Kodowanie fazowe i sprzęt potrzebny do odczytu

Pamięć kasetowa PK-1 zapisuje informacje z gęstością 32 bity na milimetr (na jednej ścieżce), z szybkością 12,7 cm/s. Zapis informacji doknywany jest metodą przemagnesowania nośnika magnetycznego taśmy przez głowicę zapisu. Wykorzystywana jest mteoda kodowania fazowego, tzw. PE (ang. Phase Encoded), polegająca na tym, że:

— bit „1" określony jest przez zmianę kierunku strumienia magnetycznego w kierunku zgodnym z polaryzacją magnetyczną przerwy międzyblokowej; zaś bit „0" w kierunku przeciwnym do kierunku tej polaryzacji

— dodatkowe zmiany kierunku strumienia magnetycznego doknywane są w środku pomiedzy zmianami strumienia zdefiniowanymi powyżej; zapis i odczyt informacji kodowanej w ten sposób pokazany został na rysunku 1.

Dane zorganizowane są w bloki, które oddzielone są przerwami międzyblokowymi. Przerwa międzyblokowa jest skasowanym odcinkiem taśmy, którego długość zawiera się pomiędzy 17,8 a 250 mm (typ. 20,3 mm). Blok danych składa się z:

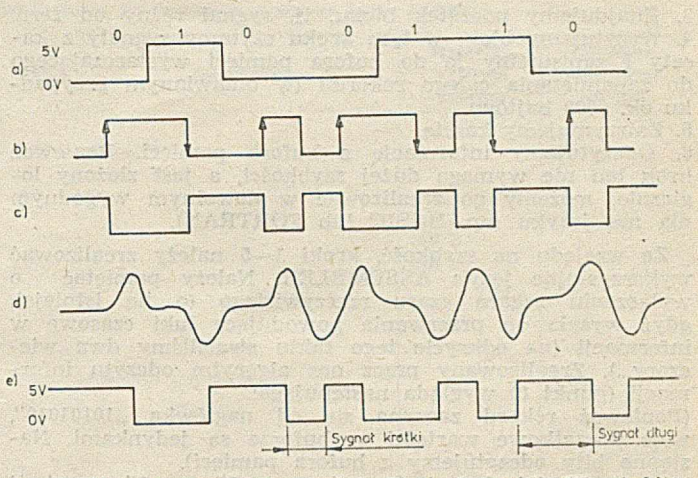
— nagłówka, tzn. ciągu bitów „10101010"

— ciągu bajtów danych; w zasadzie bajt reprezentowany jest przez 8 bitów danych + bit parzystości (nie zawsze występuje); pierwszy bit jest najmniej znaczący, ostatni jest bit parzystości

— dwóch bajtów cyklicznej kontroli nadmiarowej CRC (ang. Cyclic Redundancy Check), ale można się nimi nie przejmować, bo i tak nie będziemy ich dekodować

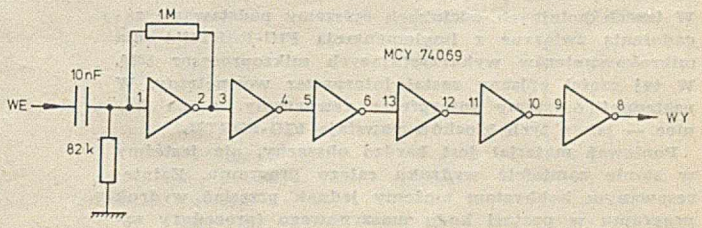
— zakończenia — tego samego ciągu bitów, co w nagłówku.

Dla uproszczenia odczytu zaleca się zapisywanie rekordów o tej samej długości, np. 128 bitów. Rzecz jasna, podany opis jest poważnie uproszczony. Pełny standard zapisu przedstawia norma: ISO/TC97/SC11 Project 6.



Rys. 1. Zapis i odczyt informacji kodowanej metodą PE: a) bity zapisanej informacji, b) informacja zakodowana, c) stan magnetyczny taśmy; d) sygnał z magnetofonu; e) uformowany sygnał odczytu

Tak zapisaną informację można odczytywać za pomocą zwykłego magnetofonu kasetowego. Na wyjściu „głośnik dodatkowy" uzyskujemy sygnał jak na rysunku 1d, z tym że należy uwzględnić różnice szybkości magnetofonu (4,75 cm/s) i stacji PK-1. W pierwszym rzędzie trzeba uzyskać uformowany sygnał odczytu (rys. 1e). Niektóre mikrokomputery, np. COMMODORE-64, VIC-20. NEPTUN-184, kodują informacje zapisywane na kasecie w analogiczny sposób; tak więc możemy czytać kasetę bezpośrednio na magnetofonie komputera. Dla innych mikrokomputerów np. ZX-81, SPECTRUM, TRS-80, należy sygnał uformować i odczytać poprzez port.



Rys. 2. Schemat opisywanego układu

Użyty przez nas układ podany został na rysunku 2. Różni się on od układów używanych w sprzęcie profesjonalnym (jest o wiele prostszy). Wprawdzie daje większą liczbę błędów odczytu (1 błąd na 10000 bajtów, bo w domu włącza się zamrażarka...), ale dla omawianych celów wystarcza w zupełności. W ten sposób uzyskujemy na wejściu komputera ciąg impulsów o poziomach 0 lub 1 i o różnych długościach, które następnie dekodujemy programowo. Jest to przykład zastąpienia odpowiednim oprogramowaniem rozwiązań sprzętowych, często dużo droższych.

Algorytm odczytu

Nie możemy niestety podać konkretnego programu odczytującego informację z kasety, gdyż zależy on od rodzaju komputera i jego oprogramowania (w naszym przypadku TRS-80 + dysk + translator FORTRANU firmy MICROSOFT). Opiszemy natomiast nasze podejście do odczytania kaset PK-1. Podstawowym założeniem było ograniczenie programowania w języku ASSMEBLER.

Odczyt jednego rekordu z kasety można podzielić na następujące kroki:

1. Odczekujemy aż wczytany bit będzie jedynką
2. Znajdujemy przerwę międzyblokową, tj. długi odcinek taśmy zawierający same zera (np. 200 zer).

3. Znajdujemy początek bloku, tj. sygnał różny od zera.
4. Wczytujemy blok; w tym kroku czytamy sygnały z kasety i zapisujemy je do bufora pamięci wystarczającego do zapamiętania całego rekordu (w omawianym przypadku ok. 9000 bajtów).
5. Zatrzymujemy kasetę.
6. Odczytujemy informację z bufora pamięci. Ponieważ krok ten nie wymaga dużej szybkości, a jest złożony logicznie, możemy go zrealizować w dowolnym wygodnym dla nas języku (np. BASIC lub FORTRAN).

Ze względu na szybkość, kroki 1—5 należy zrealizować wykorzystując język ASSEMBLER. Należy pamiętać o wyłączeniu zegara czasu rzeczywistego (o ile istnieje), gdyż wysyła on przerwania powodujące luki czasowe w informacji (na odkrycie tego faktu straciliśmy dwa wieczory...). Zrealizowany przez nas algorytm odczytu informacji (punkt 6) wygląda następująco:
(Ponieważ rekord zaczyna się od nagłówka „10101010", więc początkowe wartości w buforze są jedynkami. Następne bity odczytujemy z bufora pamięci).
a) przeszukaj bufor aż do zmiany z „1" na „0"; przyjmij za wartość wczytaną zero
b) przeskocz ustaloną liczbę miejsc w buforze (wartość ta zależy od szybkości procesora oraz algorytmu wczytywania sygnału z kasety i należy ją dobrać eksperymentalnie; w naszym przypadku wynosiła ona 6): odczytana wartość odpowiada bitowi informacji
c) czytaj znaki z bufora aż do zmiany odczytywnej wartości
d) idź do punktu b.
W ten sposób wczytujemy tyle bitów, ile powinien zawierać rekord.

W naszym przypadku rekordy nie zawierały bitów kontroli parzystości i poprawność odczytu kontrolowaliśmy poprzez dwukrotne odczytanie kasety, a następnie porównanie wyników. Dekodowanie CRC nie było konieczne.

W opisanej metodzie wystarcza minimalna ilość sprzętu, natomiast wymaga ona sporego wysiłku programisty. Zanim uzyskaliśmy końcowe wyniki, napisaliśmy serię programów testowych, rysujących na ekranie monitora odczytane przebiegi sygnału, badające rozrzut długości sygnałów długich i krótkich, co pozwoliło na dobranie odpowiednich stałych czasowych w programach. Wysiłek sowicie się opłacił, gdyż uzyskaliśmy w ten sposób bogate źródło programów, których nigdy byśmy sami nie przepisali.

Na koniec dodajmy, że bez pomocy pp. Jacka Staszelisa i Marka Kałużnego z CAMK-u przedstawiona metoda nie mogłaby być zrealizowana.

ALEKSY E. BARTNIK
MACIEJ GÓRSKI

„This public domain publication is provided through the courtesy of the Forth Interest Group, P.O. Box 1105, San Carlos, CA 94070. Further distribution must include this notice."

Poniższa publikacja stała się możliwa dzięki grupie zwolenników języka FORTH, którzy rozpowszechnili metodę zainstalowania języka „FIG-FORTH" (nieco odmiennej wersji standardu FORTH 77), obligując jedynie do zamieszczania powyższej notatki.

W trzech kolejnych odcinkach opiszemy podstawowe zagadnienia związane z implementacją FIG-FORTH'A dla mikrokomputerów wykorzystujących mikroprocesor 8080. W tej części opisany został interpreter wewnętrzny. W następnej opiszemy interpreter zewnętrzny, a na koniec — jak z tych klocków powstaje FIG-FORTH.

Ponieważ materiał jest bardzo obszerny, nie jesteśmy w stanie zamieścić wydruku całego programu. Zainteresowanym hobbystom możemy jednak przesłać wydruk programu w postaci kodu maszynowego (procedury zależne od sprzętu w postaci źródłowej — co ułatwi adaptację programu). W tym celu prosimy o przesłanie do redakcji:
— dużej zaadresowanej koperty ze znaczkiem
— rogu okładki z napisem mikroKLAN 8
— krótkiej informacji o posiadanym sprzęcie mikrokomputerowym

Kod maszynowy będzie przesyłany w wersji wymagającej ciągłego obszaru pamięci RAM od adresu 100_H do 2000_H.

Jak zainstalować FORTH (1)

Interpreter wewnętrzny i stosy języka FORTH

Dla języka FORTH charakterystycznych jest pięć elementów: słownik, stosy (danych i powrotów), interpretery (wewnętrzny i zewnętrzny), asembler oraz pamięć wirtualna. Żaden z tych elementów nie jest unikalną własnością języka FORTH; jednak ich współistnienie powoduje wzajemne potęgowanie działania, tworząc system programowy o niespodziewanej mocy i elastyczności. Omówimy stosy (danych i powrotów) oraz interpreter wewnętrzny.

FORTH korzysta z hierarchicznego systemu procedur. Na samym dole hierarchii znajdują się procedury napisane w kodzie maszynowym. Procedury stojące wyżej nie zawierają kodu maszynowego, lecz składają się z wywołań procedur podrzędnych. Liczba poziomów jest teoretycznie nieograniczona. Dowolny program napisany w języku FORTH jest zamieniany (kompilowany) w procedurę znajdującą się na szczycie pewnej hierarchicznej piramidy wywołań innych procedur. Wywołania procedur muszą się wiązać z zapamiętywaniem adresów powrotu. Ze względu na wielopoziomowość hierarchii, najbardziej naturalnym sposobem organizacji jest stos, nazywany stosem powrotów.

Oprócz stosu powrotów istnieje drugi, zupełnie odrębny — stos danych. Jest on miejscem przechowywania argumentów i wyników operacji wykonywanych przez procedury.

W przypadku wykorzystania mikroprocesora 8080, jeden ze stosów można obsługiwać poprzez użycie rozkazów CALL i RET bądź PUSH i POP. Drugi stos trzeba obsługiwać poprzez bardziej rozbudowane procedury programowe. Ostateczną postać interpretera (wersja e) przedstawimy jako wynik kolejnych kroków optymalizujących rozwiązanie.

W wersji **a** procedura wyższego poziomu składa się z kolejnych wywołań procedur podrzędnych (niekoniecznie najniższego poziomu). Użycie rozkazu CALL w procedurze nadrzędnej i RET w procedurach podrzędnych świadczy o sprzętowej obsłudze stosu powrotów. Wadą rozwiązania jest nieekonomiczne wykorzystanie pamięci; duża część (jedna trzecia) procedury nadrzędnej — kody rozkazów CALL — nie niesie istotnej informacji.

W wersji **b** z procedury wyższego poziomu wyeliminowano kody rozkazów CALL; zostały tylko adresy wywoływanych procedur podrzędnych. Takiego kodu nie może wykonywać procesor; trzeba było dodać interpreter **NEXT**. Używa on zmiennej **ip** (wskaźnik interpretacji — ang. interpretive pointer), wskazującej adresy kolejnych procedur podrzędnych i zwiększane przed każdym wywołaniem procedury o liczbę bajtów zajmowanych przez adres. Wyrażenie **[ip]** oznacza wartość umieszczoną w pamięci pod adresem wskazywanym przez **ip**. Takie rozwiązanie oszczędza pamięć i zwalnia stos sprzętowy. Nie organizuje ono jednak stosu programowego, wskutek czego hierarchiczna struktura procedur ma tylko dwa poziomy: poziom procedur nadrzędnych (będących sekwencjami adresów) i poziom procedur podrzędnych, napisanych w kodzie maszynowym.

Wersja **c** realizuje programowo stos powrotów. Jeśli kolejny element listy tworzącej procedurę nadrzędną jest adresem procedury najniższego poziomu, to interpretacja

przebiega analogicznie jak w poprzednim przypadku. Jeśli jednak element listy jest adresem procedury będącej także listą, to następuje skok do rozkazu **jmp DOCOL**, umieszczonego na początku tej listy, i wykonanie procedury **DOCOL**. Procedura ta realizuje programową obsługę stosu powrotów, umieszczając bieżącą wartość wskaźnika interpretacji w miejscu pamięci wskazywanym przez zmienną **rsp** (wskaźnik stosu powrotów — ang. return stack pointer). Odzyskanie ze stosu powrotów wskaźnika interpretacji następuje po obsłudze ostatniego adresu listy przez wykonanie procedury SEMIS. Taki sposób organizacji listy nosi nazwę bezpośredniego kodu nizanego (ang. direct threaded code).

Wersja **d** jest realizacją pośredniego kodu nizanego (ang. indirect threaded code). Wyeliminowano tu kod instrukcji **jmp**, pozostawiając tylko adres procedury **DOCOL**. Podobnie w procedurach najniższego poziomu przed pierwszą instrukcją maszynową umieszczono adres wskazujący tę instrukcję. Interpreter NEXT musi teraz zapewnić przekazanie sterowania nie na początek procedury, lecz pod adres umieszczony na początku procedury. Wskaźnik interpretacji **ip** wskazuje więc kolejny adres z listy, zaś zawartość spod tego adresu, po umieszczeniu w rejestrze **w**, wskazuje na początkowy adres umieszczony w procedurze podrzędnej, czyli adres procedury DOCOL albo adres pierwszego rozkazu procedury najniższego poziomu.

Interpreter NEXT może też działać z post-inkrementacją (wersja **e**). Nastąpiła tu zamiana kolejności zapisania rejestru **w** i zwiększenia wskaźnika **ip**. Przy wykonywaniu procedury podrzędnej, wskaźnik interpretacji przechowywany na stosie wskazuje już następną procedurę podrzędną.

```
;   NEXT - INTERPRETER ADRESOW FORTHA
;          (WERSJA Z POSTINKREMENTACJA)
;
;       REJESTRY:
;       FORTH       8080
;       -----       ----
;        IP         BC
;        W          DE
;        SP         SP
;
ORG     0143H
DPUSH:  PUSH        D       ;***STOS <- DE***
HPUSH:  PUSH        H       ;***STOS <- HL***
NEXT:   LDAX        B       ;***DE <- (BC) +1 BC <- BC+2
                            ; JMP DE***
        INX         B
        MOV         L,A
        LDAX        B       ; HL <- (BC)
        INX         B       ; BC <- BC+2
        MOV         H,A
        MOV         E,M
        INX         H
        MOV         D,M     ; DE <- (HL), HL <- HL+1
        XCHG                ; DE <-> HL
        PCHL                ; JMP HL
;
ORG     0128H
RPP:    DW          0       ; WSKAZNIK STOSU POWROTOW
;
ORG     0447H
SEMIS:  DW          $+2     ; ***BC <- RSTOS***
        LHLD        RPP
        MOV         C,M
        INX         H
        MOV         B,M
        INX         H       ; BC <- (RPP)
        SHLD        RPP     ; RPP <- RPP+2
        JMP         NEXT
;
ORG     0611H
DOCOL:  LHLD        RPP     ; ***RSTOS <- BC, DE <- DE+1
                            ;      BC <- DE+1***
        DCX         H
        MOV         M,B
        DCX         H
        MOV         M,C     ; (RPP) <- BC
        SHLD        RPP     ; RPP <- RPP-2
        INX         D       ; DE <- DE+1
        MOV         C,E     ; BC <- DE
        MOV         B,D
        JMP         NEXT
```

Wydruk 1. Realizacja interpretera wewnętrznego języka FORTH dla 8080

Na wydruku 1 przedstawiono sposób realizacji interpretera wewnętrznego (NEXT, SEMIS i DOCOL) dla mikroprocesora 8080.

Optymalizacja interpretera wewnętrznego języka FORTH

Wersja	Interpreter	Procedura wyższego poziomu	Procedura najniższego poziomu
a	procesor	. . . call A call B call C . . .	A: . . . . return
b	NEXT: ip←ip+2 jmp [ip]	. . . A B C	A: . . . jmp NEXT
c	NEXT: ip←ip+2 jmp [ip] SEMIS: ip←[rsp] rsp←rsp+2 jmp NEXT DOCOL: rsp←rsp−2 [rsp]←ip ip←[ip]+2 jmp NEXT	jmp DOCOL A B C . . . SEMIS	A: . . . jmp NEXT
d	NEXT: ip←ip+2 w←[ip] jmp [w] SEMIS: $+2 ip←[rsp] rsp←rsp+2 jmp NEXT DOCOL: rsp←rsp−2 [rsp]←ip ip←w+2 jmp NEXT	DOCOL A B C . . . SEMIS	A: $+2 . . . jmp NEXT
e	NEXT: w←[ip] ip←ip+2 jmp[w] SEMIS: $+2 ip←[rsp] rsp←rsp+2 jmp NEXT DOCOL: rsp←rsp−2 [rsp]←ip ip←w+2 jmp NEXT	DOCOL A B C . . . SEMIS	A: $+2 . . . jmp NEXT

Komunikację operatora z systemem (poprzez konsolę i drukarkę) organizują dwie procedury najniższego poziomu — KEY i PEMIT (wydruk 2). Wywołują one procedury CIN, COUT i POUT — wymieniające znaki między rejestrami procesora a konsolą i drukarką. Etykietom CIN, COUT i POUT należy przyporządkować adresy wynikające z konkretnej realizacji wykorzystywanego systemu mikroprocesorowego.

```
CIN     EQU     OF803H  ; ADR. PROC. WCZYT. ZNAK DO A
                        ; Z KONSOLI
COUT    EQU     OF809H  ; ADR. PROC. WYSYL. ZNAK Z C
                        ; NA KONSOLE
POUT    EQU     OF80FH  ; ADR. PROC. WYSYL. ZNAK Z C
                        ; NA DRUKARKE
;
ORG     1643H
EPRINT: DW      0       ; ZMIENNA STERUJACA DRUKARKA
                        ; 0=WYLACZONA 1=WLACZONA
;
ORG     02F2H
KEY:    DW      $+2     ; ***CZYTAJ ZNAK Z KONSOLI NA
                        ;     STOS. JEZELI CTRL P,
                        ;     ZMIEN STAN DRUKARKI***
        JMP     PKEY
ORG     1686H
PKEY:   CALL    CIN     ; WCZYTAJ DO A ZNAK Z KONSOLI
        CPI     10H     ; GDY ZNAK = CTRL P
        MOV     E,A     ; E <- ZNAK
        JNZ     PKEY1
        LXI     H,EPRINT;
        MVI     E,20H   ; E <- SPACJA
        MOV     A,M
        XRI     1
        MOV     M,A     ; ZMIEN STAN DRUKARKI
PKEY1:  MOV     L,E     ;
        MVI     H,0     ; HL <- 0 I ZNAK
        JMP     HPUSH   ; ZNAK NA STOS, POTEM NEXT
;
ORG     169EH
PEMIT:  DW      $+2     ; ***WYSLIJ ZNAK ZE STOSU NA
                        ;     KONSOLE***
        POP     H       ; L <- KOD ZNAKU ZE STOSU
        PUSH    B       ; CHRON IP
        MOV     C,L
        CALL    CPOUT   ; WYSLIJ NA KONS. ZNAK Z C
        POP     B       ; ODZYSKAJ IP
        JMP     NEXT
ORG     1667H
CPOUT:  CALL    COUT    ; ***WYSLIJ ZNAK Z C NA KONSOLE
                        ;     I BYC MOZE NA DRUKARKE***
        XCHG
        LXI     H,EPRINT
        MOV     A,M     ; GDY (EPRINT) <> 0
        ORA     A
        JZ      CPOU1
        MOV     C,E
        CALL    POUT    ; WYSLIJ ZNAK Z C NA DRK.
CPOU1:  RET
```

Wydruk 2. Realizacja procedur KEY i PEMIT

Na wydruku 3 przedstawiono następnych pięć procedur najniższego poziomu.

```
ORG 0156H
LIT:    DW      $+2     ; ***STOS <- (BC), BC <- BC+2***
        LDAX    B
        INX     B
        MOV     L,A
        LDAX    B
        INX     B       ; HL <-<- (BC)
        MOV     H,A     ; BC <- BC+2
        JMP     HPUSH   ; STOS <- HL
;
ORG     017AH
BRAN:   DW      $+2     ; ***BC <- BC + (BC)***
BRAN1:  MOV     H,B
        MOV     L,C     ; HL <- BC
        MOV     E,M
        INX     H
        MOV     D,M
        DCX     H       ; DE <- (HL)
        DAD     D       ; HL <- HL+DE
        MOV     C,L
        MOV     B,H     ; BC <- HL
        JMP     NEXT
;
ORG     04C9H
PLUS:   DW      $+2     ; ***STOS + STOS -> STOS***
        POP     D
        POP     H
        DAD     D
        JMP     HPUSH
;
ORG     0549H
SWAP:   DW      $+2     ; ***ZAMIEN MIEJSCAMI DWA
                        ;     OSTATNIE NA STOSIE***
        POP     H
        XTHL
        JMP     HPUSH
;
ORG     0556H
DUP:    DW      $+2     ; ***POWIEL OSTATNI NA STOSIE***
        POP     H
        PUSH    H
        JMP     HPUSH
```

Wydruk 3. Realizacja niektórych słów języka FORTH jako procedur na najniższym poziomie

KEY — czeka na wciśnięcie klawisza, po czym umieszcza na stosie 16-bitową liczbę, której bardziej znaczący bajt jest zerem, zaś mniej znaczący bajt — kodem wciśniętego klawisza. W przypadku jednoczesnego wciśnięcia klawiszy CTRL i P zmienia stan podłączenia drukarki.

PEMIT — zdejmuje ze stosu 16-bitową liczbę i wysyła mniej znaczący bajt (znak ASCII) na konsolę (i ewentualnie na drukarkę, jeśli jest przydzielona).

LIT — umieszcza na stosie 2-bajtową liczbę wskazywaną przez wskaźnik interpretacji, czyli znajdującą się na liście bezpośrednio za odwołaniem do LIT, po czym zwiększa wskaźnik interpretacji o 2, aby wskazywał kolejny adres na liście.

BRAN (BRANCH) — do wskaźnika interpretacji dodaje 2-bajtową liczbę (dodatnią lub ujemną) wskazywaną przez niego, czyli znajdującą się na liście bezpośrednio za odwołaniem do BRANCH; tym samym realizuje skok.

PLUS — zdejmuje ze stosu dwie 16-bitowe liczby, dodaje je arytmetycznie i wynik umieszcza na stosie.

SWAP — zamienia miejscami dwie 16-bitowe liczby na szczycie stosu.

DUP — powiela na stosie liczbę znajdującą się na jego szczycie.

```
ORG     00060H
CRLF:                   ; ***WYSLIJ CR I LF***
        DW      DOCOL   ; STOS PUSTY
        DW      LIT
        DW      0000DH  ; STOS: 000D
        DW      PEMIT   ; WYSLIJ 0D - STOS PUSTY
        DW      LIT
        DW      0000AH  ; STOS: 000A
        DW      PEMIT   ; WYSLIJ 0A - STOS PUSTY
        DW      SEMIS
;
LOOP:                   ; ***PETLA NIESKONCZONA***
                        ; WYGLAD STOSU
        DW      CRLF    ; -
        DW      KEY     ; KODN
        DW      DUP     ; KODN KODN
        DW      LIT
        DW      1       ; KODN KODN 0001
        DW      PLUS    ; KODN KODN+1
        DW      SWAP    ; KODN+1 KODN
        DW      DUP     ; KODN+1 KODN KODN
        DW      LIT
        DW      -1      ; KODN+1 KODN KODN FFFF
        DW      PLUS    ; KODN+1 KODN KODN-1
        DW      CRLF
        DW      PEMIT   ; KODN+1 KODN
        DW      CRLF
        DW      PEMIT   ; KODN+1
        DW      CRLF
        DW      PEMIT   ; -
        DW      BRAN    ;   SKOCZ NA POCZATEK PETLI
        DW      -36     ;   CZYLI ZMNIEJSZ IP O 36D
;
ORG 00050H
INIT:   LXI     SP,3B00H        ; INICJUJ STOS DANYCH
        LXI     H,3BA0H
        SHLD    RPP             ; INICJUJ STOS POWROTOW
        LXI     B,LOOP          ; INICJUJ WSK. INTERPR.
        JMP     NEXT            ; ROZPOCZNIJ INTERPRET.
```

Wydruk 4. Program sprawdzający

Proponujemy sprawdzenie działania interpretera wewnętrznego poprzez uruchomienie programu LOOP. Jest on napisany jako procedura wyższego poziomu (lista adresów) i działa w nieskończonej pętli. W czasie jednego przebiegu pętli program czeka na wciśnięcie klawisza z konsoli, po czym wysyła na konsolę w trzech kolejnych linijkach trzy znaki ASCII, których kody różnią się o 1, zaś środkowy jest znakiem wciśniętym na klawiaturze. Program wywołuje zarówno procedury najniższego poziomu, jak i procedurę wyższego poziomu (listę) — CRLF.

Cały FORTH (przedstawiono tylko jego fragment) potrzebuje pamięci o adresach od **0100H** do **3FFFH**, czyli ok. 16 KB. Przeprowadzając relokację programu można go umieścić i w innym obszarze pamięci. Wykonanie należy rozpocząć poprzez skok do procedury inicjującej INIT.

ZBIGNIEW POJMAŃSKI
PIE — Warszawa

Bądź realistą!
Mikrokomputer
Ci pomoże

Toniemy w powodzi informacji. Z odsieczą przychodzi jednak nowoczesna technika. Wystarczy mikrokomputer i zręcznie zastosowana baza danych, aby główny księgowy nie musiał odchorowywać (lub co gorsza odsiadywać) każdego bilansu, aby magazynier naprawdę wiedział co posiada, a szary petent nie przeklinał biurokraty, który za każdym razem żąda dziesiątek zaświadczeń — bo nie jest w stanie odszukać dokumentów sprzed miesięca. Prawdę mówiąc, nie potrafię sobie wyobrazić firmy, w której nie istniałaby potrzeba zastosowania tego typu oprogramowania! I to nie dlatego, żeby pani Iksińska mogła zbijać bąki — gdy komputer za nią pracuje, lecz by móc wykorzystać posiadane informacje w tak efektywny sposób, że nawet pięć zaharowanych pań Iksińskich nie byłoby w stanie tego przygotować. To może trudne do uwierzenia, ale opisywane narzędzie jest dostępne u nas w kraju — za złotówki!

BANK DANYCH-CSK

Bank Danych — CSK to system zarządzenia relacyjną bazą danych opracowany przez Computer Studio Kajkowski. Może on zostać wykorzystany dla mikrokomputerów zbudowanych na bazie mikroprocesorów 8-bitowych 8080 i Z80 lub 16-bitowego 8086. W oparciu o Bank Danych — CSK możliwe jest tworzenie różnych systemów użytkowych, takich jak:
— systemy finansowo-księgowe
— systemy płacowe
— systemy osobowe, bibliograficzne itp.
— systemy magazynowe
— systemy biurowe.

Wymagania sprzętowe

Mikrokomputer, na którym implementowany jest Bank Danych — CSK musi być wyposażony w system operacyjny CP/M (dla mikrokomputera 16-bitowego — CP/M-86) lub system z nim kompatybilny, a ponadto — dysponować przynajmniej 48 KB pamięci operacyjnej. W skład wymaganej minimalnej konfiguracji sprzętowej wchodzi: monitor alfanumeryczny i klawiatura, co najmniej jedna jednostka pamięci dyskowej oraz drukarka. System może pracować m.in. na takich mikrokomputerach, jak: ELWRO 513, IMP 85, RTDS-8, LIDIA oraz ROBOTRON 5110 i 5120.

Właściwości użytkowe systemu

System składa się z pakietu programów zapisanych na dyskietce w kodzie maszynowym mikroprocesora 8080 (8086). Ma on interpreter własnego języka manipulacji plikami (ponad 60 instrukcji i funkcji). Język ten umożliwia działania na plikach, wykonywanie obliczeń numerycznych, utrzymanie plików danych, tworzenie raportów, zestawień itp. Instrukcje tego języka mogą być wykonywane zarówno w trybie konwersacyjnym, jak i programowym. Wprowadzanie i wyprowadzanie danych może odbywać się w dostosowanej do potrzeb użytkownika postaci graficznej.

W systemie zastosowano relacyjny model bazy danych, w którym pliki składają się z rekordów o jednakowej — definiowanej przez użytkownika — długości i strukturze. Rozwiązanie takie wybrano ze względu na prosty i wygodny dla użytkownika sposób opisu zbiorów.

Wielkość plików danych, przetwarzanych w systemie, ograniczona jest następującymi parametrami:
— liczba rekordów w jednym pliku nie przekracza 65 535
— liczba znaków w rekordzie nie przekracza 1000
— liczba pól w rekordzie nie przekracza 32
— liczba znaków w polu nie przekracza 254
— liczba znaków klucza indeksowego nie przekracza 100.
Pola rekordu mogą być typu alfanumerycznego, numerycznego lub logicznego.

Zakładanie plików danych

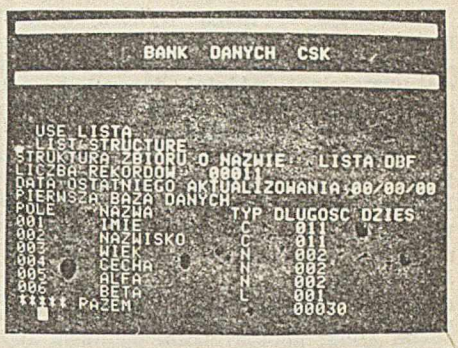
Podstawową instrukcją umożliwiającą zakładanie plików danych jest **CREATE.** Po wprowadzeniu tej instrukcji, użytkownik podaje dalej w trybie konwersacyjnym (odpowiadając na pytania) nazwę tworzonego pliku i strukturę rekordu. Struktura rekordu definiowana jest przez podanie opisów kolejnych pól, składających się z nazwy pola, typu i długości (wyrażonej liczbą znaków alfanumerycznych). W przypadku pola numerycznego możliwe jest zadeklarowanie liczby miejsc po przecinku dziesiętnym (wtedy długość wyraża sumaryczny rozmiar pola wraz z pozycją przecinka). Definiowanie struktury rekordu kończy się po wciśnięciu w odpowiedzi na kolejne pytanie o nazwę pola — klawisza CR.

Opis struktury pliku zapisywany jest na dyskietce i można go odtworzyć podając następujące instrukcje:

USE <nazwa pliku>

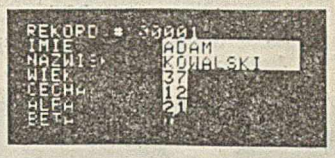
LIST STRUCTURE

USE zapewnia otwarcie dostępu do pliku o podanej nazwie, a LIST STRUCTURE powoduje wyświetlenie struktury rekordu danego pliku. Przykład działania tych instrukcji pokazano na zdjęciu 1. Plik o nazwie LISTA liczy jedenaście rekordów, z których każdy zawiera sześć pól o nazwach: IMIE, NAZWISKO, WIEK, CECHA, ALFA, BETA. Początkowe dwa pola są typu alfanumerycznego i mają długość jedenastu znaków. Następne trzy — to pola numeryczne o długości dwóch znaków; ostatnie jest polem zawierającym wartości logiczne o długości jednego znaku.



Fot. 1. Opis struktury przykładowego pliku danych

Po zakończeniu definiowania struktury pliku, użytkownik musi odpowiedzieć na pytanie, czy chce od razu wprowadzać dane. Gdy podana zostanie odpowiedź twierdząca (tak) — system wyświetla obraz pustego rekordu w standardowej postaci, oczekując na wprowadzenie danych. Na zdjęciu 2 pokazano przykładowy obraz z monitora po wprowadzeniu danych.



Fot. 2. Standardowa postać rekordu generowana przez system

Wciśnięcie klawisza CR spowoduje teraz przejście do wypełniania danymi następnego rekordu. Wprowadzanie danych kończy się wciśnięciem klawisza CR bezpośrednio po wyświetleniu kolejnego pustego rekordu. Utworzony plik zostaje zapisany w pamięci zewnętrznej.

Zakładanie plików danych można także realizować za pomocą instrukcji:

COPY — umożliwia kopiowanie całych plików lub wybranych rekordów spełniających określony warunek; kopiować można rekordy lub wybrane pola

JOIN — łączy dwa pliki danych, tworząc trzeci; również w tym przypadku do nowo tworzonego pliku można przenosić określone rekordy i pola rekordów łączonych plików.

Zakładając pliki danych, można korzystać z plików utworzonych za pomocą innych systemów lub języków programowania. Umożliwiają to odpowiednie opcje instrukcji tworzących pliki.

Aktualizacja plików danych

Dopisywanie danych do plików realizują instrukcje:

APPEND — dopisuje nowe rekordy na koniec pliku

INSERT — wpisuje nowe rekordy w dowolne miejsce pliku.

Instrukcje te mogą wykorzystywać standardową postać rekordu (jak na zdjęciu 2) lub dopisywać puste rekordy wypełniane później danymi przez odpowiedni program użytkownika.
Poprawianie i aktualizację założonych już plików umożliwiają instrukcje:

EDIT — wprowadza zmiany do określonego rekordu

BROWSE — poprawia na ekranie monitora jednocześnie kilkanaście rekordów

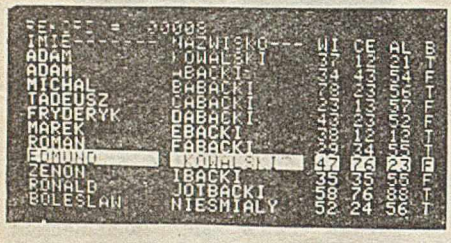
CHANGE — zmienia zawartość określonego pola w rekordzie

DELETE — zaznacza określone rekordy jako skreślone

PACK — fizycznie usuwa z pliku skreślone rekordy

UPDATE — aktualizuje plik danych na podstawie pliku zmian zapisanego w pamięci zewnętrznej.

Instrukcje EDIT i BROWSE wykorzystują możliwości edycji pełnoekranowej. Za pomocą odpowiednich klawiszy można sterować położeniem kursora i umieszczać go w dowolnym miejscu ekranu. Podobnie można zmieniać numer poprawianego rekordu. Dzięki temu, użytkownik może szybko przeglądać dane i redagować je w dowolnej kolejności.



Fot. 3. Działanie instrukcji BROWSE

Na zdjęciu 3 pokazano działanie instrukcji BROWSE. Kursor umieszczono na pozycji rekordu nr 8, który przez to wyświetlony został w odmienny sposób. Umieszczając teraz kursor na żądanym znaku można dokonać stosownych zmian.

Wybrane możliwości systemu

Oprócz wspomnianych wcześniej możliwości, system ma wiele innych instrukcji. Są to m.in.:

SORT — sortowanie pliku według określonego pola danych

INDEX — indeksowanie pliku danych

TOTAL — scalanie plików danych poprzez sumowanie rekordów o tej samej zawartości określonych pól

LOCATE — wyszukiwanie rekordów spełniających podany warunek logiczny

FIND — szybkie odnajdywanie rekordu na podstawie klucza indeksowego

LIST — drukowanie plików danych (lub ich części spełniających określony warunek logiczny) w postaci całych rekordów lub wybranych pól

REPORT — szybkie tworzenie standardowych zestawień z możliwością umieszczania w nich nie tylko danych z pliku, lecz również wyników obliczeń

COUNT — obliczanie liczby rekordów spełniających podany warunek

SUM — sumowanie wybranych pól danych rekordów spełniających określony warunek

SET ALTERNATION — tworzenie dziennika dostępu do plików.

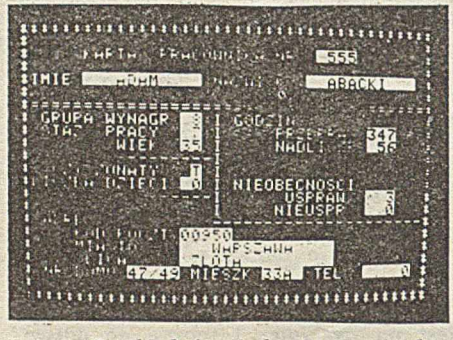
Podane instrukcje można wykonywać zarówno w trybie konwersacyjnym, jak i programowym.

Tworzenie systemów użytkowych

Bogaty repertuar instrukcji Banku Danych — CSK umożliwia tworzenie systemów użytkowych. Mogą być one pisane za pomocą dowolnego systemu redagowania tekstów. Można również do tego celu wykorzystać instrukcję MODIFY COMMAND, która w istocie dostarcza możliwości prostego systemu tego typu.

Programy użytkowe mogą korzystać ze zmiennych i stałych oraz kilkunastu funkcji numerycznych, logicznych i działających na ciągach znaków. Obliczenia numeryczne przeprowadzane są z dokładnością do dziesięciu cyfr.

Wśród instrukcji umożliwiających tworzenie programów, oprócz instrukcji warunkowych typu IF...ELSE czy organizujących pętle programowe DO WHILE..., znaleźć można takie, które pozwalają wprowadzać i wyprowadzać dane w dogodnej dla użytkownika postaci. Można zatem korzystać ze standardowej postaci rekordu, generowanej przez system, lub też przedstawić ją w innej formie na ekranie monitora — np. jak na zdjęciu 4.



Fot. 4. Przykład innej formy wprowadzania danych

Pola danych dla większej czytelności wyświetlane są w odmienny sposób od komunikatów. Wprowadzane dane mogą być na bieżąco kontrolowane przez program użytkowy. Podobnie, do wyprowadzania danych można użyć instrukcji LIST, REPORT lub też — używając innych dostępnych instrukcji — zaprojektować własną formę wydruku.

Tworząc konkretny system użytkowy, można go zrealizować w takiej postaci, że korzystanie z niego nie będzie wymagało umiejętności programowania (może być obsługiwany przez osoby nie przygotowane zawodowo do pracy z komputerem) i będzie on odporny na błędy użytkownika.

Przedstawiony system cechuje prostota opisu danych i duża uniwersalność zastosowanego języka. Walory użytkowe prezentowanego systemu zwiększa dodatkowo możliwość jego współpracy z innymi pakietami programów o szerokiej gamie zastosowań, np. TABPLAN-CSK — systemem kalkulacyjno-planistycznym lub TEKST-CSK — systemem redagowania tekstów.

MACIEJ GILARSKI
Computer Studio Kajkowski

- Ogłoszony przez ABAKUS pierwszy w Polsce konkurs na grę edukacyjną dobiegł końca. Zgłoszono 25 programów. Pierwszą nagrodę (ZX SPECTRUM 48K), zdobył Wojciech Dąbrowski za program „Rysowanie" (zabawa dla dzieci kilku- i kilkunastoletnich, ucząca myślenia algorytmicznego przy rysowaniu).
  Przyznano także trzy wyróżnienia:
  — Krzysztofowi Konieczce za program „GEO" (odgadywanie położenia geograficznego miast polskich na mapie)
  — Barbarze Chrzan-Feluch (poprzednio laureatka w konkursie na pomysł) za program „Kostka wielowymiarowa" (wyrabiający orientację w przestrzeniach wielowymiarowych)
  — Markowi Wojciechowskiemu za program „Nauka żeglowania".
  Jury odrzuciło wszystkie programy napisane na MERĘ — jako że żaden z autorów nie spełnił warunku dostarczenia komputera dla demonstracji działania programu.
  Konkurs obnażył... brak werwy w pisaniu i nadsyłaniu programów, także u ludzi z fantazją. Kilka niezłych pomysłów nie zostało dopracowanych; i to zarówno koncepcyjnie, jak i pod względem programowym. Uwaga ta odnosi się również do laureatów.
- PPZ AMEPROD wydało drugi numer „ZX MIKRO". Trochę o sprzęcie, trochę programów — w sumie godne uwagi. Cena 250 zł, dość wysoka dla prywatnego nabywcy, wynika z kosztów druku i opracowania.
- Uwaga właściciele COMMODORE C64. Bartosz Pastuszak (66-400 Gorzów Wielkopolski, ul. Marcinkowskiego 10g m. 3, tel. 26-401 lub 25-151) chętnie wymieni doświadczenia z inymi posiadaczami tego mikrokomputera.
- Nadsyłając przez najbliższy miesiąc pod adresem INFORMATYKI: zaadresowaną do siebie dużą kopertę ze znaczkiem, róg odkładki pisma z napisem „mikro-KLAN 8", ewentualną ocenę dotychczasowych publikacji o ZX SPECTRUM — można otrzymać kserokopię artykułu (w języku angielskim) zawierającego procedury w kodzie maszynowym, ułatwiające wykorzystanie możliwości graficznych ZX SPECTRUM.

— prowadzi
Andrzej J. Pietrowski
tel. dom. 48-22-85